

Security Ontology for Annotating Resources

Anya Kim, Jim Luo, and Myong Kang

Center for High Assurance Computer Systems,
Naval Research Laboratory, Washington, DC 20375
{kim, lu, mkang}@itd.nrl.navy.mil

Abstract. Annotation with security-related metadata enables discovery of resources that meet security requirements. This paper presents the NRL Security Ontology, which complements existing ontologies in other domains that focus on annotation of functional aspects of resources. Types of security information that could be described include mechanisms, protocols, objectives, algorithms, and credentials in various levels of detail and specificity. The NRL Security Ontology is more comprehensive and better organized than existing security ontologies. It is capable of representing more types of security statements and can be applied to any electronic resource. The class hierarchy of the ontology makes it both easy to use and intuitive to extend. We applied this ontology to a Service Oriented Architecture to annotate security aspects of Web service descriptions and queries. A refined matching algorithm was developed to perform requirement-capability matchmaking that takes into account not only the ontology concepts, but also the properties of the concepts.

1 Introduction

In today's network-centric computing environment, automatic discovery of resources and the ability to share information and services across different domains are important capabilities [1]. The first step in providing these capabilities is to markup these resources with various metadata in a well-understood and consistent manner. Such annotation will enable resources to be machine-readable and machine-understandable.

Using metadata to find distributed resources that meet one's functional requirements is only the first step. Resource requestors may have additional requirements such as security, survivability, or quality of service (QoS) specifications. For example, they may require resources to possess a certain military classification level, to originate from trusted sources, or to be handled according to a specified privacy policy. Therefore, resources need to be sufficiently annotated with security-related metadata so that they can be correctly discovered, compared, and invoked according to security as well as functional requirements of the requestor.

In this paper, we introduce a set of security-related ontologies collectively referred to as the NRL Security Ontology. The NRL Security Ontology provides the ability for precisely describing security concepts at various levels of detail. This ontology complements existing ontologies that mainly focus on functional aspects of capability, content, and parameters. Marking up security aspects of resources is a crucial step toward deploying a secure Service Oriented Architecture (SOA) system.

Other groups have recognized the need for security annotation of services and proposed a set of security-related ontologies [2-4]. However, these ontologies possess certain limitations discussed in Section 2. The NRL Security Ontology was created to address these limitations. We expect this work to serve as a catalyst in the development of standardized security-related ontologies with contributions from both the security community and the semantic Web community.

The rest of the paper is organized as follows. Section 2 examines previous work in security ontology and discusses the need for improvements. Section 3 presents the NRL security ontology, including design objectives, domain and scope, and detailed descriptions. Section 4 gives examples of how to use these ontologies to annotate and query for resources particularly in a Web service context. It also discusses our algorithm for matchmaking between queries and resource descriptions. Section 5 presents future work and our conclusion.

2 Existing Security-Related Ontologies

Realization of the need for security ontologies is not new. Denker et al. have created several ontologies for specifying security-related information in Web services [2] using Daml+OIL [5] and later OWL [6]. We refer to this set of ontologies as the DAML Security Ontology for the rest of the paper. The authors state that the goal of these ontologies is to enable high-level markup of Web resources, services, and agents, while providing a layer of abstraction on top of various Web service security standards such as XML-Enc [7], XML-Dsig [8], and SAML (Security Assertion Markup Language) [9].

Of the set of ontologies that make up the DAML security ontology, the two main ontologies are the Security Mechanisms ontology and the Credential ontology. They describe security mechanisms and authentication credentials respectively. While we realize that these ontologies are works-in-progress and provide a great foundation for describing security-related concepts, we found two issues with them. First, they are not intuitive to understand especially in terms of the organization of subclass relationships. Second, they cannot express all the security information that we want to describe or be easily extended to do so.

The intuitiveness issue is particularly true for the main Security Mechanisms ontology. Figure 1 depicts this ontology in a simplified form where circles denote classes, solid lines represent instances of the classes and dotted lines represent properties¹. The top class in this ontology is 'SecurityMechanism' with subclasses of 'SecurityNotation', 'Signature', 'Protocol', 'KeyFormat', 'Encryption', and 'Syntax'. Making these unrelated concepts sibling classes does not make sense from either a security perspective or an ontology perspective. Furthermore, some instances are not properly assigned to the correct subclass. For example, Kerberos and SSH are both declared as instances of 'KeyProtocol', however these are not key protocols. Additionally, all properties are defined for the top class. However, those properties do not apply to most of the subclasses. For example, no instance under the 'Syntax' subclass would have a need for the *relSecNotation* (Relative Security Notation), *enc* (Encryption), *sig* (Signature), or *reqCredential* (Required Credential) properties, yet they are all inherited because these properties are defined at the top class.

¹ Some complex concepts they use such as restriction classes are not depicted here.

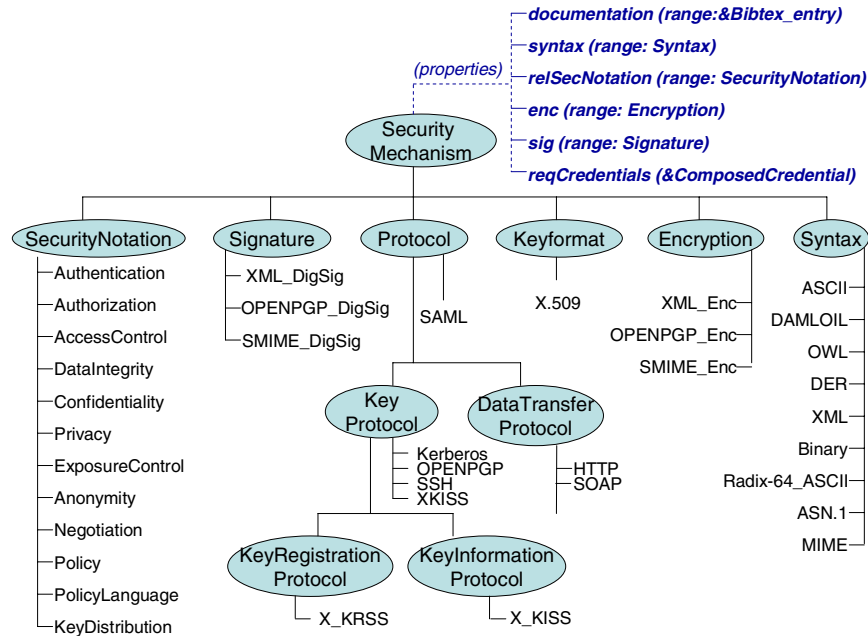


Fig. 1. Simplified DAML Security Mechanisms Ontology

The second issue we mentioned is the lack of expressiveness. The DAML security ontology includes many classes and instances that are not directly relevant for security annotation while lacking others that are necessary. For example, syntax and data transfer protocols are useful concepts in another domain, but are not particularly relevant for describing security-related information. Furthermore, the only encryption instances defined in the ontology are S/MIME, OpenPGP, and XML encryption. We do realize that more instances could be added as the need arises. However, the organization of the class hierarchy should be well developed. For example, there should be classes to represent military as well as commercial security devices and security policies. Currently, there is no appropriate place in the DAML Security Ontology to create a firewall or military security policy instance. There is also a lack of appropriately placed properties that could allow for more detailed refinement of security concepts. For example, it would be useful to define the algorithms supported by a protocol, or the certification status of a mechanism.

Although the authors of the DAML Security Ontology did a great job in recognizing the need for security ontologies and beginning work in security ontologies, we feel that there is still room for improvement. The next section describes the NRL Security Ontology in detail.

3 NRL Security Ontology

The DAML Security Ontology focuses on annotation of Web services rather than resources in general. This is evident not only from their documentation [2], but also

by examining the types of classes and instances in the ontology. We want ontologies that can be used to annotate generic resources from simple documents to interactive services with security-related metadata. We also want to improve upon the limitations of the DAML Security Ontology outlined in the previous section. The NRL Security Ontology was designed with the following objectives in mind:

1. Describe security related information applicable to all types of resources
2. Provide the ability to annotate security related information in various levels of detail for various environments (both commercial and military)
3. Create ontologies that are easy to extend and provide reusability
4. Facilitate mapping of higher-level (mission-level) security requirements to lower-level (resource-level) capabilities

3.1 Domain and Scope of the Ontology

When creating an ontology, one of the most important factors is the domain and scope in which it will be used [10]. While our objectives outlined above are a good starting point, in order to create ontologies that will be truly useful, we need to understand the types of questions that the ontology will be expected to answer.

These ontologies will be used by both the resource provider and the requestor to express their security requirements and capabilities. We must consider the various ways that the same statement can be expressed. Furthermore, we need to consider statements that are unlikely in order to limit the scope of the ontology. Statements that are either too broad or too specific are unlikely to be used and provide no useful information.

Noy et al. [10] state that one of the best ways to determine the scope of the ontology is to list a set of *competency questions* that can be answered using the ontology. For our purposes we did the same by composing a list of security requirements and capabilities for both the resource requestor and the provider. From the requestor's perspective, security requirements can be stated in terms of specific mechanisms or in terms of abstract security objectives. From the resource provider's perspective, security requirements are similar to the notion of policy and can express concepts such as authentication and access control. The provider's capabilities include protocols and mechanisms that the provider possesses and security policies it adheres to. The actual list of the requirements and capabilities statements we created can be found in the extended version of this paper [11].

3.2 Organizational Structure of NRL Security Ontology

We chose OWL to create our ontologies because it provides a rich vocabulary for describing classes and properties [6, 12]. It is widely used in many communities that have begun to develop ontologies of their own knowledge domains [13].

There are seven separate ontologies that make up the NRL Security Ontology:

1. Main Security ontology: an ontology to describe security concepts
2. Credentials ontology: an ontology to specify authentication credentials
3. Security Algorithms ontology: an ontology to describe various security algorithms
4. Security Assurance ontology: an ontology to specify different assurance standards
5. Service Security ontology: an ontology to facilitate security annotation of semantic Web services

6. Agent Security ontology: an ontology to enable querying of security information
7. Information Object ontology: an ontology to describe security of input and output parameters of Web services

The Service Security, Agent Security, and Information Object ontologies are based on some existing DAML Security ontologies while the others are new. The Credentials, Security Algorithms, and Security Assurance ontologies provide values for properties defined for concepts in the Main Security ontology. They enable those concepts to be described in more detail with respect to types of credentials used, supported algorithms, and associated levels of assurance. The Service Security ontology provides the means to use security concepts from the Main Security ontology in the Web services framework. The Agent Service ontology enables creation of security-related queries using security concepts from the Main Security ontology. The Information Object ontology allows for annotation of Web service inputs and outputs using the Security Algorithms ontology. The relationship among these ontologies is represented in Figure 2. The ontology depicted in gray represents OWL-S, a set of core ontologies used to describe Web services.

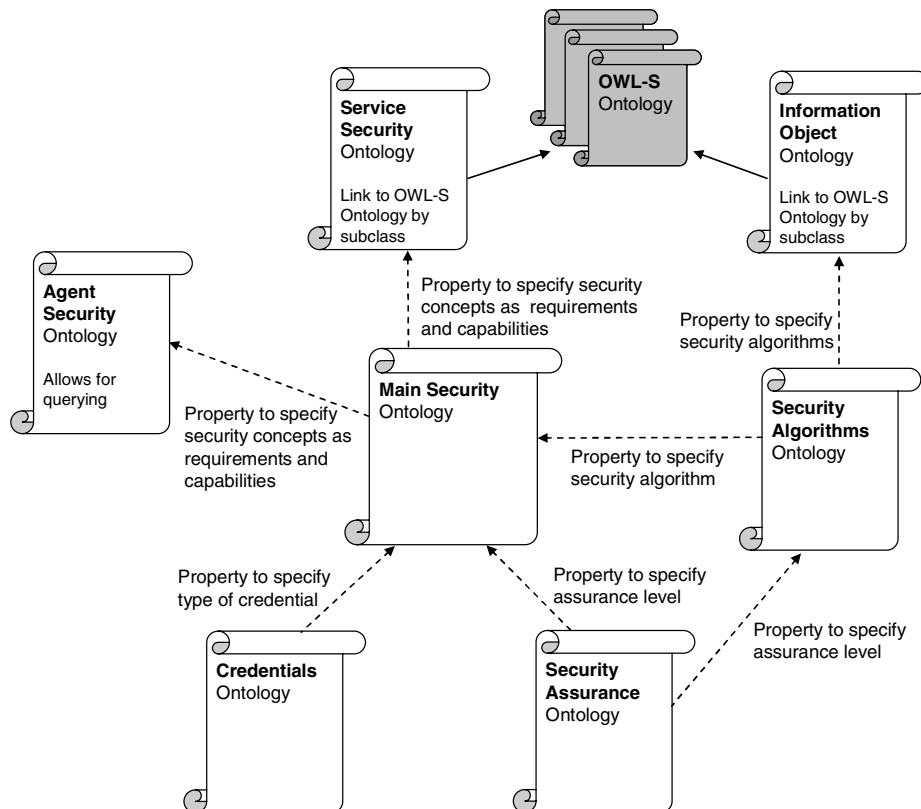


Fig. 2. Graphical Representation of Security-Related Ontologies and Their Relationships

Next, we present a brief explanation of classes, properties and relationships in each ontology. Due to space limitations we do not show all ontologies here. A complete graphical depiction of these ontologies and the OWL files can be found in [11].

Main Security Ontology (securityMain.owl). The core ontology in the NRL security ontology set is the Main Security ontology (Figure 3). It imports the Credentials ontology, Security Algorithms ontology, and Security Assurance ontology as object properties. The top class, ‘SecurityConcept’ possesses three subclasses: ‘SecurityProtocol’, ‘SecurityMechanism’ and ‘SecurityPolicy’.

While some may argue that the distinction between security protocols and security mechanisms is blurred, we define security protocols as an agreed upon series of steps to accomplish a task while security mechanisms are implementations of protocols [14]. We specifically differentiate them here to provide the ability to describe security in both manners. Security policies are the set of rules that regulate how information is protected and secured .

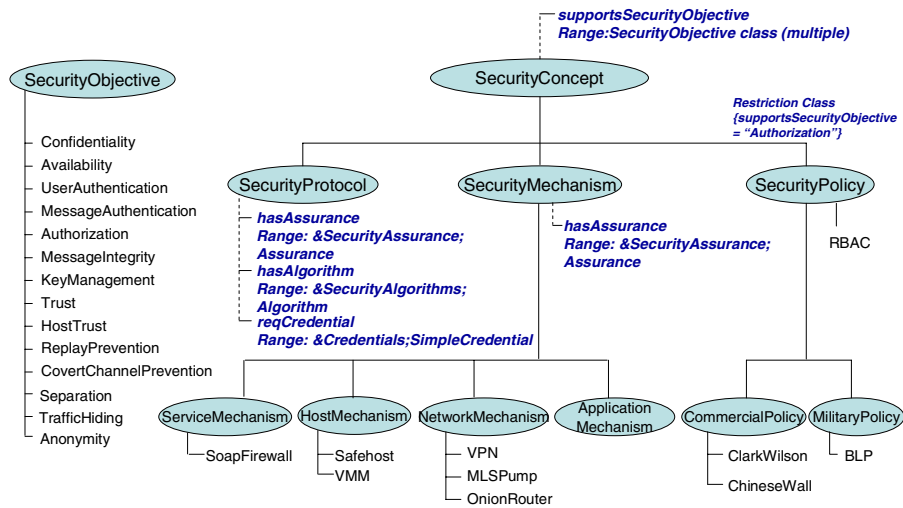


Fig. 3. A Part of the Main Security Ontology

The Main Security ontology also has a separate class called ‘SecurityObjective’ that enables users to specify security objectives for the ‘SecurityConcept’ class using the *supportsSecurityObjective* property. For example, IPSec is declared to have Confidentiality, MessageAuthentication, and TrafficHiding as its *supportsSecurityObjective* property values. Security objectives also enable users to search for protocols, mechanisms, or policies based on the security objective they require. For example, users can query, “find all instances that provide confidentiality” and receive a list of all the security concepts that have a value of Confidentiality in their *supportsSecurityObjective* property.

Another way we can use ‘SecurityObjective’ is to map high-level mission requirements to low-level service requirements. For instance, assume that a security requirement is specified at the mission level such that Mission 1 and Mission 2 must have separation between them. At this level, the mission planner can use the ontology to specify the security objective of Separation. The mission designer can then search for instances in the ‘SecurityConcept’ class that provide Separation. In this case, the only one that does is VPN, so he can select VPN as a security requirement at the service level.

Credentials Ontology (credentials.owl). Authentication is one of the most fundamental security requirements in a networked environment. The Credentials ontology allows for specification of credentials used for authentication purposes (Figure 4). Concepts in the Security Main ontology can refer to a specific credential through their *reqCredential* property. While we adopted some of the notations in the DAML Credential ontology, we improved upon it by reorganizing classes to be more intuitive, including more properties and adding more classes to define additional types of credentials. Our Credentials ontology categorizes credentials into physical token, electronic token, and biometric token.

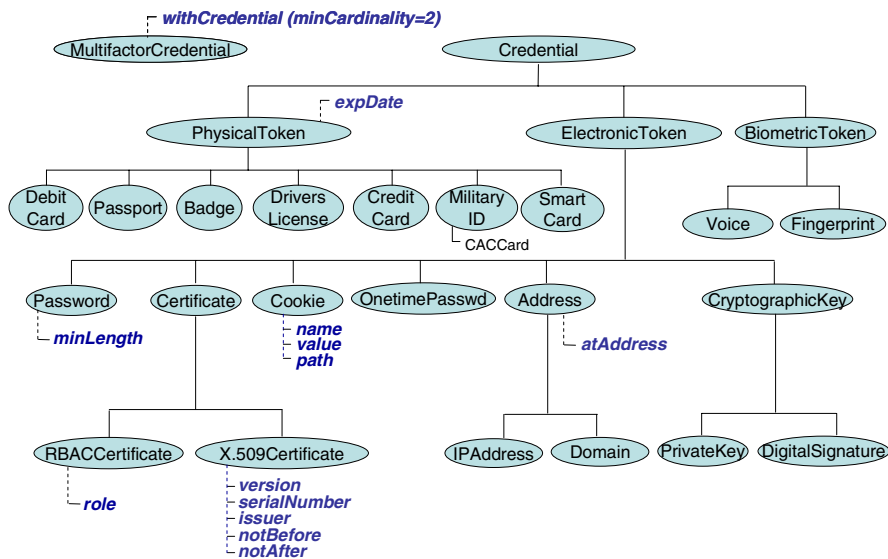


Fig. 4. Credentials Ontology

Under the ‘PhysicalToken’ class, we kept many of the classes from the DAML Credential ontology under their ‘IDCard’ class. In addition, we created a class to describe military IDs and an instance to represent CAC (Common Access Card) cards used in the military. The ontology can be extended to add properties such as issuing agency, expiration date, issue date, etc. Under the ‘ElectronicToken’ class, we

provide subclasses that enable authentication based on host address, certificates, passwords, and cryptographic keys to name a few. Additional properties were added to describe certificates including the issuer, version and serial number under the Certificate class. In order to support role-based (RBAC) certificates [15], an ‘RBACCertificate’ class was created as a subclass of the Certificate class with a *role* property. The ‘BiometricToken’ class represents credentials that pertain to human traits. For now, only ‘Voice’ and ‘Fingerprint’ subclasses are defined here.

In addition to the three categories of simple credentials, the ‘MultifactorCredential’ class can be used to describe composed credentials made up of two or more individual credentials. For example, it can describe requirements where both a smart card as well as a password is needed.

Security Algorithms Ontology (securityAlgorithms.owl). The Security Algorithms ontology was created to enable description of various security algorithms (Figure 5).

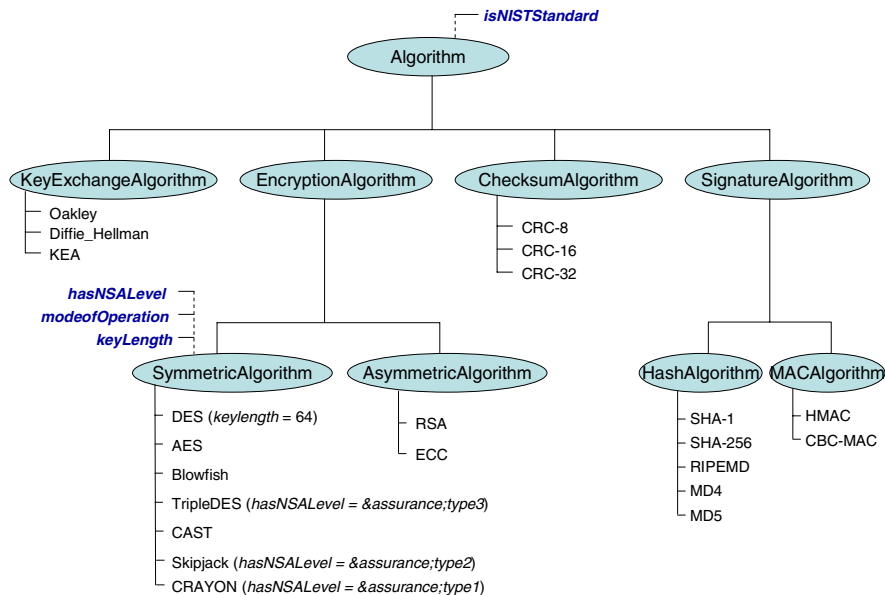


Fig. 5. Security Algorithms Ontology

Security Assurance Ontology (securityAssurance.owl). The Security Assurance ontology provides a way to describe standardized assurance methods for security protocols, mechanisms, and algorithms. They can be described in terms of their assurance level using the *hasAssurance* property from the Main Security ontology. The ‘Assurance’ class is classified according to different assurance methods: ‘Standard’, ‘Accreditation’, ‘Evaluation’, and ‘Certification’. This ontology is the least complete of all our ontologies. However, we have added classes to describe the Common Criteria and TCSEC evaluations, and the FIPS and NSA standards [16].

Service Security and Agent Security Ontologies (serviceSecurity.owl and agentSecurity.owl). OWL-S [17] is an OWL-based semantic markup description language that provides a core set of constructs for describing Web services specifically. It provides a set of ontologies called Profile, Process, and Grounding to describe Web services. The Profile describes services in terms of what the service does, the Process describes how to use it, and the Grounding specifies how to interact with it.

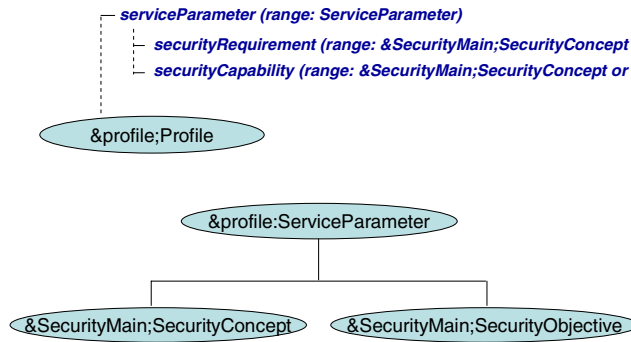


Fig. 6. Service Security Ontology

In order for the NRL Security Ontology to be used in the Web service context, a link must be made to the OWL-S ontologies. The Service Security ontology was developed for such a purpose. In the Service Security Ontology, ‘SecurityConcept’ and ‘SecurityObjective’ from the Main Security ontology are defined to be subclasses of the ‘ServiceParameter’ class in the OWL-S Profile ontology (Figure 6). The OWL-S Profile also contains a *serviceParameter* property that can have ServiceParameter as its value². Declaring two subproperties of the *serviceParameter* property, *securityRequirement* and *securityCapability* enables the OWL-S Profile to include security requirements and security capabilities in its service description. Furthermore, we defined the range for these subproperties as either the ‘SecurityConcept’ or ‘SecurityObjective’ classes. This allows security requirements and capabilities to be stated in terms of either a particular security objective, or a specific security mechanism.

The Agent Security ontology allows for querying of resources, in particular Web services with requestor requirements and capabilities. It defines an ‘Agent’ class to represent the service requestor with the properties *securityCapability* and *securityRequirement* that can hold values from the ‘SecurityConcept’ and ‘SecurityObjective’ classes.

Information Object Ontology (InfObj.owl). The Information Object ontology is based on a DAML ontology created to capture encrypted or signed input/output data

² Note that the OWL-S Profile ontology has a property and class of the same name, *serviceParameter*. However, the property starts with a lowercase letter, while the class starts with an uppercase letter. Thus, *serviceParameter* refers to a property while *ServiceParameter* refers to a class.

of Web services. It has an ‘InfObj’ class and two subclasses, ‘EncInfObj’ (Encrypted Information Object) and ‘SigInfObj’ (Signed Information Object). The ‘InfObj’ class is used as the range for input and output parameters of services described with OWL-S. The ontology has the *cryptoAlgUsed* property to specify the algorithm used to encrypt or sign the object. In the original DAML ontology, the *cryptoAlgUsed* property pointed to a set of algorithms defined within the DAML Information Object ontology. However, we felt that the two concepts of information object and security algorithms were so dissimilar that they did not belong within the same ontology file. Hence, in the NRL Information Object ontology, the *cryptoAlgUsed* property points to classes in the Security Algorithms ontology.

3.3 Design Objectives Revisited

At the beginning of Section 3 we outlined a set of objectives expected to be achieved by the NRL Security Ontology. This subsection discusses whether those design objectives were met and to what degree.

1. **Describe security related information not only for Web services, but for all types of resources:** The NRL Security Ontology enables us to describe security information of various types of resources. We can describe security protocols that are specific to Web services such as XML-enc and SAML, but also include many protocols and mechanisms such as IPSec, Kerberos and SSH that are generally applied to any resource.
2. **Provide the ability to annotate security related information in various levels of detail for various environments:** The ontology can provide specific details of security mechanisms through properties such as the types of algorithms supported, required key length, types of credentials used, and expiration dates. Classes and instances were created that enable description of resources relevant to a military environment as well as for commercial use.
3. **Create ontologies that are easy to extend and provide reusability:** The ontologies are created with a class hierarchy that makes sense from a security perspective. New instances when necessary can be added to the ontology in an intuitive manner without having to alter the class hierarchy.
4. **Facilitate mapping of higher-level (mission-level) security requirements to lower-level (resource-level) capabilities using the ontology:** resources can be described in terms of either security objectives at the abstract level, or security concepts at the concrete level. A mapping was established so that moving between the two methods of specification is possible.

In the next section, we will provide some examples of how to apply these ontologies to annotate resources with security information.

4 Application of NRL Security Ontology to a Service Oriented Architecture

While the NRL Security Ontology can be used to describe security-related information of resources in general, in this section we discuss how to annotate Web services in a Service Oriented Architecture. In particular, we focus on:

- How to annotate Web service descriptions with security requirements and capabilities
- How to create queries for finding Web services with given security requirements and capabilities
- How to perform matchmaking between queries and service descriptions in the SOA context.

4.1 Reasoning and Matching Algorithm

We have stated that both resource requestors and providers have security requirements and capabilities. Matchmaking looks for a two-way correspondence between these requirements and capabilities. In other words, service requirements are compared to requestor capabilities and service capabilities are compared to requestor requirements. In order for a match of security concepts to occur between a service provider and a service requestor, two conditions should be met. First, the provider's security capabilities should satisfy the requestor's security requirements. Second, the provider's security requirements should be satisfied by the requestor's security capabilities. This implies that the requirements should subsume the capabilities (Table 1).

Table 1. The Matching between Requestor and Provider Requirements and Capabilities

Requestor		Provider
Requirements	\subseteq	Capabilities
Capabilities	\supseteq	Requirements

Every single requestor requirement must have a corresponding capability on the provider side to satisfy it, and vice versa. Hence the matchmaker must be able to perform two tasks. First, it must be able to determine the level of match for each specific requirement and a specific capability. Second, it must use those levels of match to determine if the set of requirements is matched by the set of capabilities. In other words, the matchmaker must determine the level at which each requirement is matched to a capability, and then the overall level of match between the requester and the provider. This will be explained in detail later in the section.

Several semantic matching algorithms have been proposed [2, 18, 19]. Two of these [18, 19] support only one-way matching of functional service descriptions to requestor queries as opposed to requirement-capability matching. They do not need to consider two-way matching since their focus is on matching functional aspects; when discussing purely functional requirements there is no functional requirement from the provider-side and no functional capabilities on the requestor-side. The third proposed matchmaking algorithm [2] performs requirement-capability matching for both sides. However, it does not take into account property attributes. Consequently, it will not support cases where both the requirement and capability point to the same concept but the concepts are annotated with different properties. For example, the requestor and provider may both use SSH (stated as a requirement on one side and a capability on the other), but if the requestor requires SSH using TripleDES and the provider is only capable of SSH with AES then these two should not match. Our matchmaker will perform requirement-capability matching, taking into account property annotations.

Specifically, when describing security information of resources, the ability to include properties in the matching algorithm is very important. This is due to the fact that security information, more so than functionality-related information can require detailed descriptions that make extensive use of properties. Complex statements can be made with multiple layers of properties. For example, there could be a security requirement that requires the use of XML-enc (*securityRequirement* property) with a symmetric encryption algorithm (*hasAlgorithm* property) that has been declared a type 3 algorithm from the NSA (*hasNSALevel* property).

For the first task of the matchmaker, there are four possible levels of match for each requirement-capability pair: perfect match, close match, possible match, and no match in decreasing order of matching.

Perfect Match cases. Perfect matches occur when both one's capability and the other's requirement point to the same concept. The same concept can mean the exact same concept, or two concepts declared as equivalent in the ontology. There are two ways this can occur:

- Case 1. Both the requirement and capability specify the exact same ontology concept. The instances and property values specified by both sides are identical. This is the trivial case. For example, if a requestor query states that it requires the service to possess a VPN (Virtual Private Network) that possesses a Common Criteria EAL4 rating and a service describes its capability as possessing a VPN with a Common Criteria rating of EAL 4 then these two are a perfect match.
- Case 2. The requirement and capability refer to equivalent concepts, and if properties are specified, the properties are identical or equivalent. For example, a requestor's requirement specifies SSL and the provider's capability is listed as TLS. In the Main Security ontology, these two concepts are listed as equivalent classes; hence they are identical and will produce a perfect match. We sometimes call this an equivalence match to differentiate from the first case.

Close Match cases. A close match occurs when one's requirement is more general (i.e., described in less detail) than the other's capability. There are three ways this can occur:

- Case 1. The requirement specifies a more general concept at a higher level in the ontological hierarchy. For example, the requestor's capability is stated as DES while the provider's requirement asks for a symmetric encryption algorithm. DES is an instance of the 'SymmetricAlgorithm' class and thus lower in the hierarchy. We assume that the provider specified its requirement as a higher level concept because it does not care which specific algorithm is used as long as it is a symmetric encryption algorithm. Therefore, we can assume a match.
- Case 2. The requirement and capability have the same concept, but the capability is specified in more detail (i.e., property). For example, the requestor's capability is specified as AES with 256 bit keys while the provider's requirement asks for AES (with no properties). AES with 256 bit keys is a more specific instance of AES so we can assume that there is a match.
- Case 3. The requirement is stated in terms of a security objective while the capability is stated in terms of a security concept that supports that specific objective. For example, the requestor's requirement is stated as the objective of

Confidentiality and the provider's capability is given as XML-Enc which has the *supportsSecurityObjective* value of Confidentiality. Since the requirement is looking for anything that supports Confidentiality and XML-Enc does support it, we view this as a match.

Possible Match cases. A possible match occurs when one's requirement is more specific (i.e., defined in more detail) than the other's capability. This is the opposite of a close match. A possible match does not rule out the possibility of a match, but the information available cannot ensure the capability can match the requirement. There are three ways this can occur:

- Case 1. The requirement specifies a more specific concept (lower in the hierarchy). For example, the requestor's capability is stated as symmetric encryption algorithm while the provider's requirement asks for DES. The symmetric encryption algorithm that the requestor is capable of could be DES, but it is not certain. Therefore, it is only a partial match.
- Case 2. The requirement and capability refer to the same concept, but the requirement specifies a more refined concept (i.e. property). For example, the capability is stated as AES while the requirement asks for AES with 256-bit keys. The AES specified in the capability could be possible of 256-bit key encryption, but it is not certain. Therefore, it is only a partial match.
- Case 3. The requirement is stated in terms of a security concept while the capability is stated in terms of a security objective that is supported by the security concept. For example, the requestor's requirement is stated as confidentiality while the provider's capability is stated as XML-Enc which supports confidentiality. The requestor may be capable of using XML-Enc, but it is not certain. All we can deduce is that the requestor is capable of confidentiality. Therefore, it is only a partial match.

No Match cases. No match occurs when one's capability and the other's requirement are disparate without the possibility of matching. There are two ways this can occur:

- Case 1. The requirement and capability point to two unrelated concepts. For example, the requirement states it requires DES and the capability states its capability as RSA. These concepts have no hierarchical relationship to each other and so are unrelated. There can be no match.
- Case 2. The requirement and capability point to the same concept but have different specifics (i.e. properties) with respect to that concept. For example, the requirement points to AES in CBC mode while the capability states AES in CFB mode. The capability and requirement can both use AES, but they require modes of operation; one is a block cipher the other is a stream cipher so they are not compatible.

For the second task of the matchmaker, it must attempt to match every requirement on one side against every capability on the other side. The degree of match for a single requirement is its highest level of match it has against all of the possible capabilities. The overall level of match between the requester and the provider is the same as the lowest degree of match of any of the requirement-capability pairs. There are four possibilities:

- If at least one of the requirements is not matched, then the requestor is not matched to the provider. The requestor will not be able to use the resource.
- If all the requirement-capability pairs are at least possible matches, then there is a possible match between the requester and the provider. This means there is not enough information to determine one way or the other whether the requester can use the resource. Additional information or negotiation will be needed to make that determination.
- If all the requirement-capability pairs are at least close matches, then the requestor can indeed use the resource.
- If all the requirement-capability pairs are perfect matches, then obviously the requestor can use the resource.

In the following section, we will provide an example of the matching process between a service description and a query.

4.2 Application of the Matching Algorithm

In this section we examine how to actually describe services and create queries using the security ontologies, and how to find services using the matching algorithm. In our example, we have a service requestor looking for a book selling service. The service requestor would create queries to find services that match not only the desired functionality, but also the security capabilities and requirements of the requestor.

The following is an example of the requestor's security capabilities and requirements along with the part of their query that pertains to the security capability and requirements:

Requestor's Security Capability

1. Authentication via SAML with an X.509 Certificate signed by VeriSign

Requestor's Security Requirement

1. Authorization
2. SSH with the DES algorithm in CBC mode

```
<credential:X.509Certificate rdf:ID="X.509">
  <credential:issuer rdf:resource="VeriSign"/>
</credential:X.509Certificate>
<securityMain:SAML rdf:ID="Capability1">
  <securityMain:reqCredentials
    rdf:resource="&credential;X.509"/>
</securityMain:SAML>
<securityMain:Authorization rdf:ID="Requirement1"/>
<securityAlgorithms:DES rdf:ID="Alg">
  <securityAlgorithms:modesOfOperation rdf:resource="CBC"/>
</securityAlgorithms:DES>
<securityMain:SSH rdf:ID="Requirement2">
  <securityMain:hasEncryptionAlgorithm
    rdf:resource="&securityAlgorithms;Alg1"/>
</securityMain:SSH>
<agent:Agent rdf:about="#BookRequest">
  <securityCapability rdf:resource="#Capability1"/>
  <securityRequirement rdf:resource="#Requirement1"/>
  <securityRequirement rdf:resource="#Requirement2"/>
</agent>
```

On the other hand, a book selling service would create an OWL-S profile that includes its functional capabilities, as well as security requirements and capabilities. The following is the example security capability and requirement statements of the book selling service (BookSeller), along with the part of its OWL-Profile that would contain these statements.

BookSeller's Security Capability

1. SOAP Firewall with a Common Criteria level of EAL4
2. SSH with DES

BookSeller's Security Requirement

1. Authenticate via SAML with an X.509 Certificate

```
<securityMain:SOAPFirewall rdf:ID="Capability1">
  <securityMain:hasAssurance rdf:resource="&assurance;EAL4"/>
</securityMain:SOAPFirewall>
<securityMain:SSH rdf:ID="Capability2">
  <securityMain:hasEncryptionAlgorithm
    rdf:resource="&securityAlgorithms;DES"/>
</securityMain:SSH>
<credential:X.509Certificate rdf:ID="X.509"/>
<securityMain:SAML rdf:ID="Requirement1">
  <securityMain:reqCredentials
    rdf:resource="&credential;X.509"/>
</securityMain:SAML>
<profile:Profile rdf:about="#BookSeller1">
<profile:serviceName>BookSeller1</profile:serviceName>
<profile:textDescription>
  This service sells all types of books
</profile:textDescription>
  <securityCapability rdf:resource="#Capability1"/>
  <securityCapability rdf:resource="#Capability2"/>
  <securityRequirement rdf:resource="#Requirement1"/>
</profile:Profile>
```

Given this service description and the above query, the matching algorithm would match the requestor's capabilities to the provider's requirements and the requestor's requirements to the provider's capabilities in the following manner (Tables 2 and 3):

Table 2. Matching Requestor's Capabilities to Provider's Requirements

Requestor Security Capability	Provider Security Requirement	Match Level
Authentication via SAML with an X.509 Certificate signed by VeriSign	Authentication via SAML with an X.509 Certificate	Close Match

Table 3. Matching Requestor's Requirements to Provider's Capabilities

Requestor Security Requirement	Provider Security Capability	Match Level
Authorization	SOAP Firewall with Common Criteria level EAL4	Close Match
SSH with DES algorithm in CBC mode	SSH with DES algorithm	Possible Match

- In Table 2, the requestor's capability and the provider's requirement possess the same concepts, but the capability has more detail. This is Case 2 of the close match situation.
- In the first row of Table 3, the requestor's requirement was that a service provides Authorization. While the security objective of authorization is not explicitly stated in the OWL-S Profile of the provider, the reasoner was able to deduce that the SOAP Firewall supports authorization since it has a value of Authorization in its supportsSecurityObjective property. This is Case 3 of the close match situation.
- In the second row of Table 3, the requestor has a more detailed requirement regarding SSH than the provider has specified as its capability. This is Case 2 of the possible match situation. This could mean that either the provider cannot support the CBC mode of DES or it can support DES in CBC mode but decided not to provide this additional detail.

Since the lowest level of match in the three sets of requirement-capability pairs is possible match, the matchmaker will declare the service to be a possible match. The requester is not certain whether it can use the service. It must obtain additional information or negotiate with the provider to make that decision.

5 Conclusion and Future Work

Annotating resources with metadata enables them to be machine-understandable and facilitates automatic discovery and invocation. Most work in the area thus far has focused on annotation of resources in terms of functionality. However, security is an important issue especially in a network-centric environment. Most resources on the network are protected by some sort of security mechanisms. Satisfying functional requirements alone may not guarantee access to desired resources. As a result, annotation of resources in terms of security is just as important as annotation in terms of functionality.

In this paper, we presented the NRL Security Ontology for making security annotations. It is much more comprehensive than security ontologies previously available in terms of the number of concepts, the properties of the concepts, and the type of resources that can be described. Its organization is also more intuitive so that it is easier to use as well as to extend. New properties and instances can be added without modifying the overall class hierarchy. We demonstrated how the ontology can be applied to the context of Web services in a Service Oriented Architecture to describe security capabilities and requirements. A matchmaking algorithm was presented to perform requirement-capability matchmaking that takes into account not just the concepts, but also the properties of the concept. This is important because security annotations make extensive use of property attributes. The ability to take them into account makes this matching algorithm much more refined than previous work.

The creation of these ontologies is an iterative process. Additional instances and properties will always be needed to express new security statements. Classes and properties may be added and deleted as the security community continues to evaluate and refine the security ontologies. Additional ontologies are still needed to address issues such as privacy policies, access control, survivability, and QoS. We hope this work will serve as a catalyst in the development of standardized security-related ontologies with contributions from both the security community and the semantic Web community.

References

1. IA Architecture and Technical Framework (2004). Executive Summary of the End-to-End IA Component of the GIG Integrated Architecture, National Security Agency Information Assurance Directorate.
2. Denker, G., Kagal, L., Finin, T., Paolucci, M., and Sycara, K. (2003). Security for DAML Web Services: Annotation and Matchmaking. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*: Sanibel Island, Florida.
3. Denker, G., Nguyen, S., and Ton, A. (2004). OWL-S Semantics of Security Web Services: a Case Study. In *1st European Semantic Web Symposium*: Heraklion, Greece.
4. Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., and Sycara, K. (2004). Authorization and Privacy for Semantic Web Services. In *AAAI Spring Symposium, Workshop on Semantic Web Services*: Stanford, California.
5. W3C (2001). DAML+OIL (March 2001) Reference Description, <http://www.w3.org/TR/daml+oil-reference>.
6. W3C (2004). OWL Web Ontology Language Overview, <http://www.w3.org/TR/owl-features/>.
7. IETF and W3C Working Group (2001). XML Encryption, <http://www.w3c.org/Encryption/2001>.
8. IETF and W3C Working Group (2003). XML Signature, <http://www.w3c.org/Signature>.
9. OASIS SSTC (2005). Security Assertion Markup Language (SAML) 2.0 Technical Overview, Working Draft, <http://www.oasis-open.org/committees/download.php/12938/sstc-saml-tech-overview-2.0-draft-06.pdf>.
10. Noy, N.F., and McGuinness, D.L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory, KSL-01-05.
11. Kim, A., Luo, J., and Kang, M. (2005). Security Ontology for Annotating Resources. pp. 51, Naval Research Lab, NRL Memorandum Report, NRL/MR/5540-05-641: Washington, D.C.
12. W3C Recommendation (2004). OWL Web Ontology Language Guide, vol. 2005, W3C.
13. DAML Ontology Library. <http://www.daml.org/ontologies/>.
14. Schneier, B. (1996). Applied Cryptography, 2nd Edition (New York: John Wiley and Sons, Inc.).
15. Ferraiolo, D.F., Kuhn, D.R., and Chandramouli, R. (2003). Role-Based Access Control (Norwood, MA: Artech House).
16. Committee on National Security Systems (2003). National Information Assurance (IA) Glossary. pp. 85, http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf: Ft. Meade, MD.
17. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2003). OWL-S: Semantic Markup for Web Services, <http://www.daml.org/services/owl-s/1.1/overview/>.
18. Jaeger, M., and Tang, S. (2004). Ranked Matching for Service Descriptions using DAML-S. In *Enterprise Modelling and Ontologies for Interoperability (EMOI), INTEROP 2004*: Riga, Latvia.
19. Srinivasan, N., Paolucci, M., and Sycara, K. (2004). Adding OWL-S to UDDI, Implementation and Throughput. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*: San Diego, California.